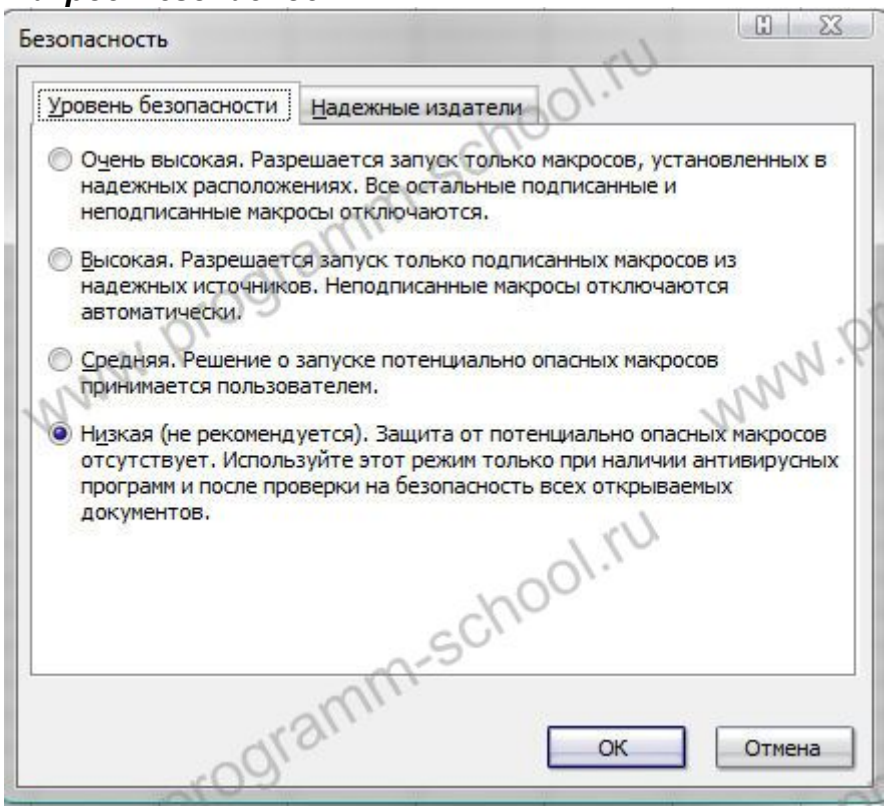
Как создать макрос в Excel?

Опубликовал Deys в вт, 21/05/2013 - 23:39

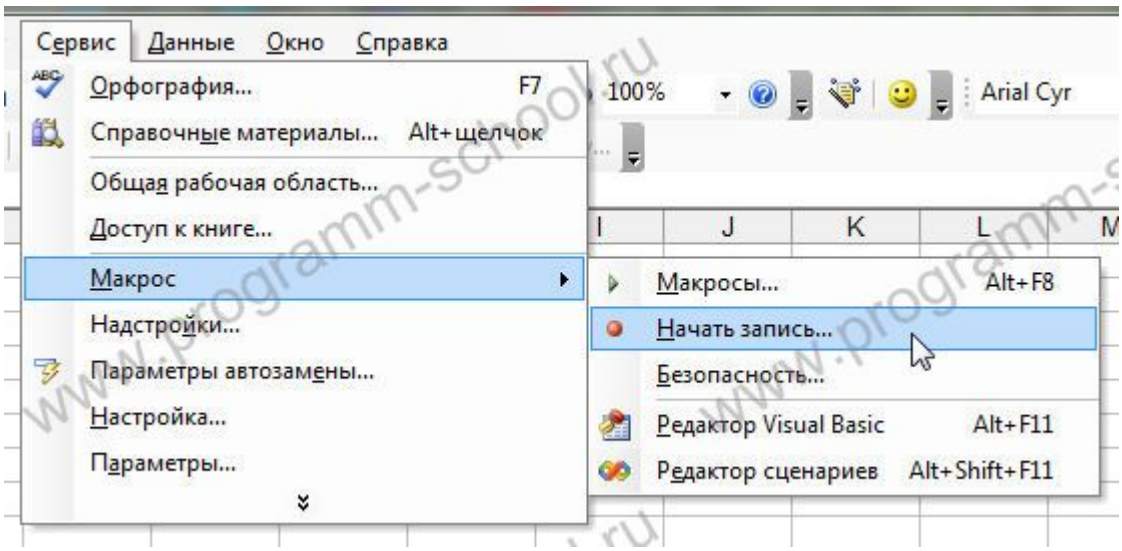
Давайте рассмотрим способы создания макросов в Excel. Первым делом Вам необходимо проверить настройку безопасности для того, что бы макросы были включены, иначе ничего не получится. Перейдите главное меню «***Сервис-Макрос-Безопасность***»



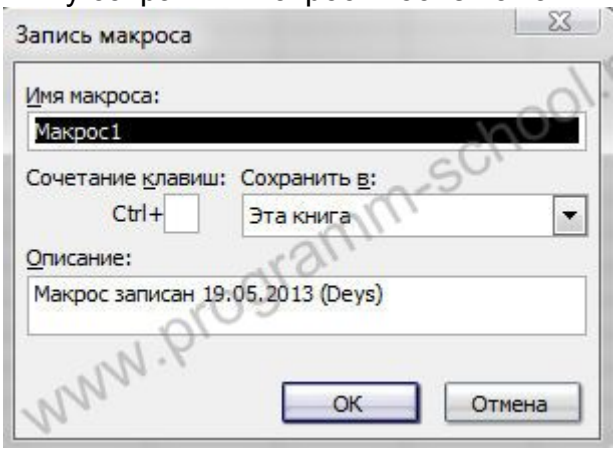
Поставьте флажок уровня на низкую (при запуске книг с макросами Excel вопросов о блокировке не задает) или среднюю (будет выдаваться предупреждение). Для учебных целей можно установить безопасность на низкую. Перезапустите Excel.
В Excel есть два способа создания макроса:

1. Записать с помощью соответствующего пункта меню
2. Создать вручную

Первый способ легкий и не требует никаких знаний в программировании. Достаточно в главном меню выбрать ***Сервис->Макрос->Начать запись…***

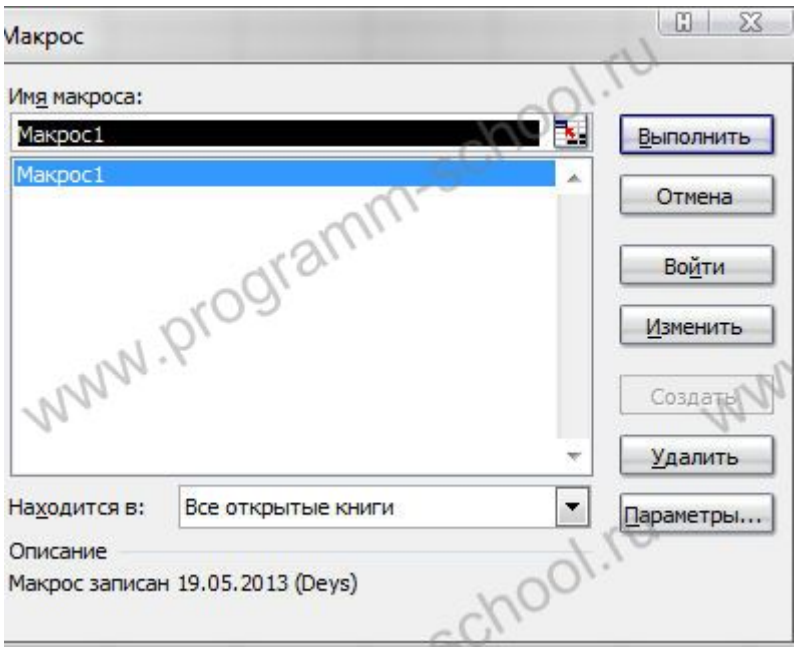


В открывшемся окне записи макроса необходимо указать его имя, которое будет выводиться в списке доступных макросов, можно добавить описание (для чего макрос, автор и т.д.), присвоить клавишу для быстрого запуска и указать в какую книгу сохранить макрос. После нажатия «**OK**» начнется запись



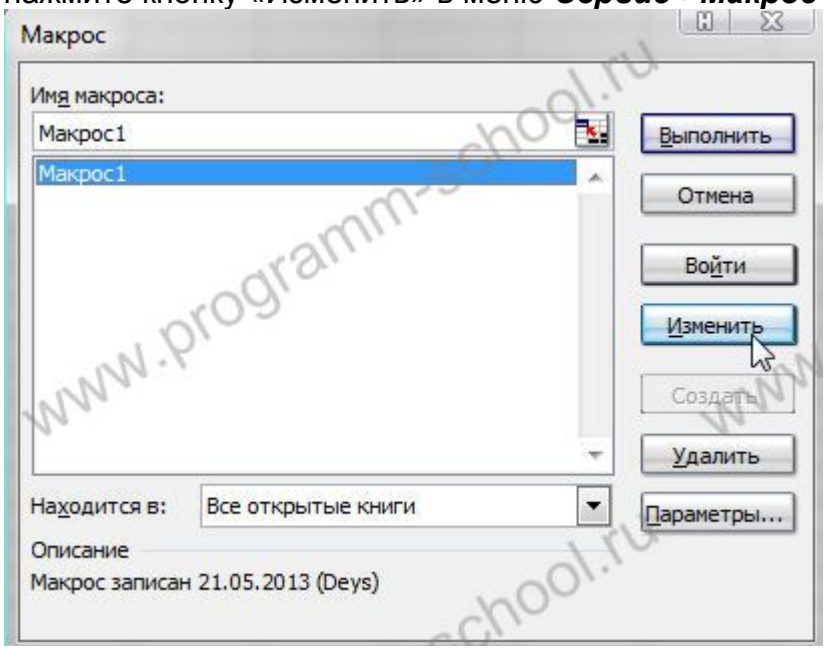
Теперь, все что Вы будете делать в рабочей книге (добавлять, изменять, удалять, создавать сводные и т.д.) все будет записываться. Для примера напишите в ячейке B3=45, B4 = 5, а в B5 формулу «=B3+B4*10». Для остановки записи необходимо нажать соответствующую кнопку:

После завершения записи наш макрос появится в списке ***Сервис->Макрос->Макросы (Alt+F8)***



Остается его только выбрать и нажать «**Выполнить**».

Все действия, которые мы произвели во время записи, с точностью повторятся. Для проверки очистите лист и выполните макрос. Но такой способ не удобен и практически в дальнейшем применить запись невозможно т.к. отсутствует универсальность. Плюс в том, что мы записывая какие либо действия получаем готовый код, который в умелых руках становится универсальным и затачивается под необходимые задачи. Давайте рассмотрим, какой код был записан. Для этого нажмите кнопку «Изменить» в меню ***Сервис->Макрос->Макросы***.



Откроется следующий код:

```
Sub Макрос1()
  Range("B3").Select
  ActiveCell.FormulaR1C1 = "45"
  Range("B4").Select
```

```
  ActiveCell.FormulaR1C1 = "5"
  Range("B5").Select
  ActiveCell.FormulaR1C1 = "=R[-2]C+R[-1]C*10"
  Range("B6").Select
End Sub
```

Sub … **End Sub** – все макросы запускаемые через меню ***Сервис->Макрос->Макросы*** начинаются с ключевого слова **Sub**(процедура). Далее следует название процедуры «Макрос1», оно же имя нашего макроса которое указывается в момент начала записи. Пустые скобки обязательны! Следует учесть, что «запускаемая» процедура не должна содержать никаких параметров, иначе макрос исчезнет из списка. Все процедуры в VB завершаются командой **End Sub**. **Sub** имеет дополнительные ключевые слова**Private** и **Public**, определяющие зону видимости процедуры. Об этом будет рассказано в следующих статьях.

Range("B3").Select – эта и последующие команды были записаны когда мы выделяли ячейки B3, B4, B5.

ActiveCell.FormulaR1C1 – команда записывающая значение или формулу в выделенную ячейку после знака равенства. Данная запись присвоения ячейке значения и формулы не очень удобна. На следующих уроках мы будем использовать свойство Cells объекта рабочего листа Worksheet.

Вот и все. Простейшие действия записаны, но вот только такую запись на практике не применить.

Второй способ, запись кода VBA вручную. Данный способ будет рассмотрен на следующем уроке и на всех последующих, будем работать только вторым способом.

Примеры макросов в Excel. Диалоговое VBA сообщение msgBox

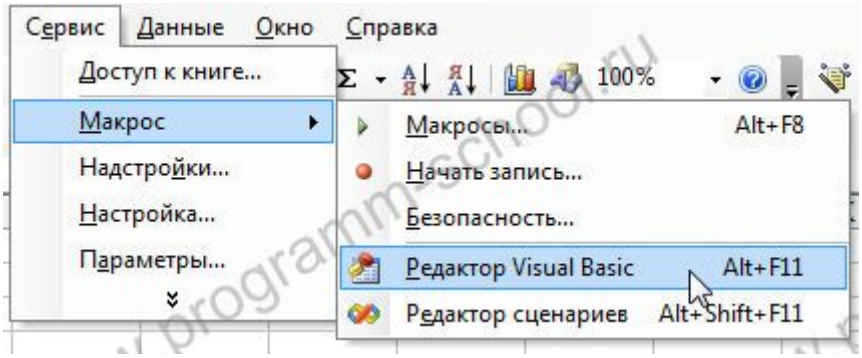
Опубликовал Deys в пт, 24/05/2013 - 00:06

В предыдущей статье мы вкратце познакомились с записью макроса при помощи рекордера Excel. Сейчас мы углубимся в эти дебри и напишем для примера пару простейших макросов. В одном из примеров по традиции продемонстрируем работу с диалоговым окном типа «Сообщение», в другом расширим наше диалоговое окно.

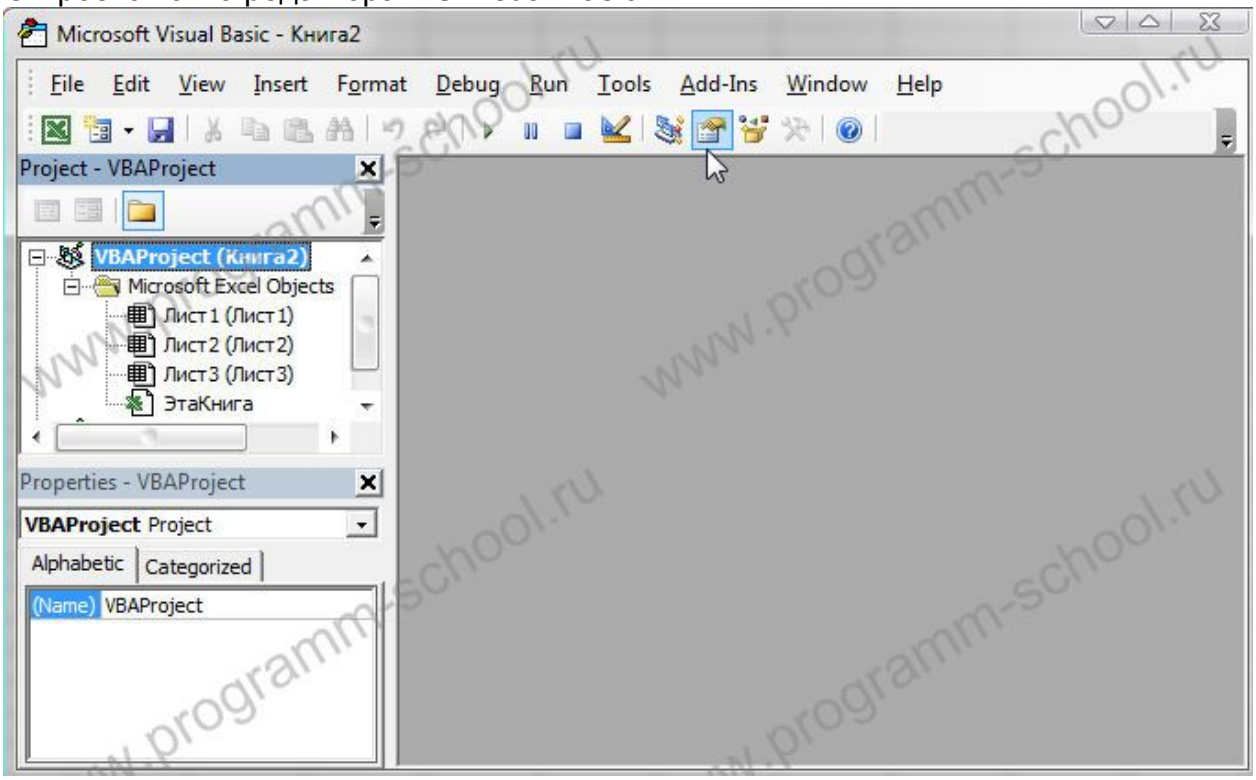
Пример 1. Простое диалоговое сообщение msgBox в VBA

Не будем отступать от традиций начала всех примеров программирования. Напишем макрос, который при запуске выдаст нам окно сообщения с надписью «Hello World». Заодно рассмотрим работу с пользовательскими диалогами.

Теперь писать макросы будем только в ручном режиме, никаких рекордеров!
Итак, для того чтоб создать макрос, Вам необходимо открыть окно всеми любимого редактора Visual Basic (VB). Для этого выполняем следующие действия: **Сервис-Макрос-Редактор Visual Basic (Alt+F11).**



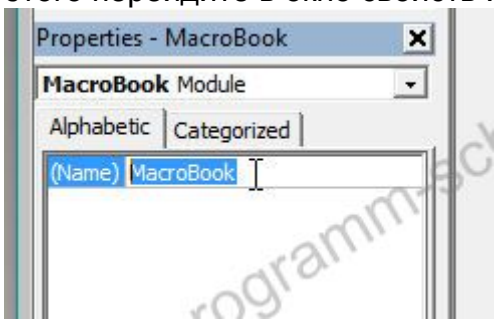
Откроется окно редактора MS Visual Basic.



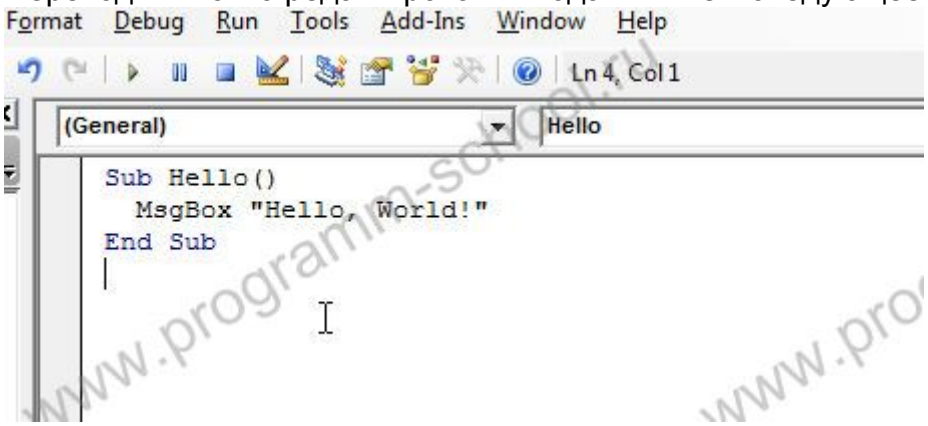
Если у Вас отсутствуют левые окна, то их необходимо включить. Для этого нажмите F4 – Открывает окно свойств **PropertiesWindow,** и сочетание клавиш Ctrl+R – открывает окно **Project Explorer.** Без этих окон в дальнейшем затруднительно работать. Все! Сделали.
Что мы видим в окне **Project**? В данном окне отражается как раз таки структура нашей книги. Объекты книги – Лист1, 2, 3, Эта книга. Более подробно изучим данные объекты в последующих статьях, а пока возвращаюсь к примеру.
Что необходимо сделать для того чтоб начать писать код? Необходимо создать модуль. *Примечание: Вообще, в дальнейшем рекомендую разделять код обработчиков на разные модули. Это облегчит понимание кода и создаст порядок в структуре.*

Создаем модуль: **Insert – Module**

Перед нами открылось пустое окно модуля, напоминает блокнот. Такое окно мы уже видели, когда записывали первый макрос в[прошлой статье](). По правилам «хорошего тона» дадим имя нашему модулю, назовем его «**MacroBook**». Для этого перейдите в окно свойств и введите имя в поле **(Name)**



Переходим в окно редактирования кода и пишем следующее:



Готово! Открываем окно рабочей книги Excel, жмем **Alt+F8** и видим наш макрос «Hello»

Остается только «**Выполнить**». В результате работы макроса мы получим сообщение следующего вида и содержания:

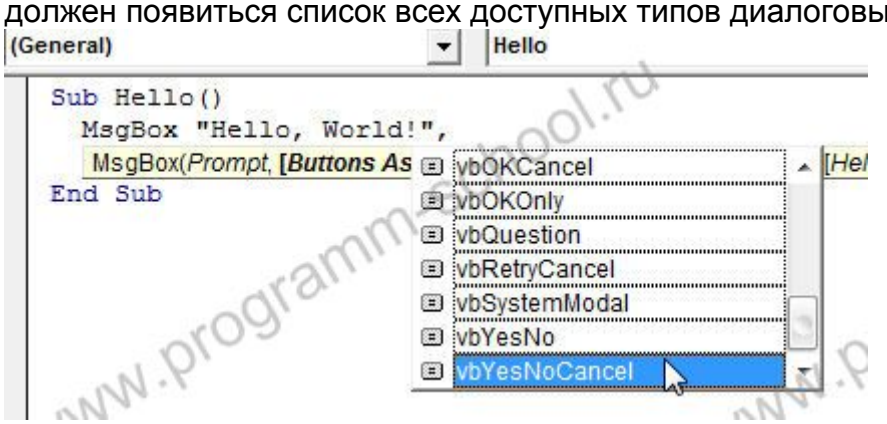
Пример 2. Расширенное диалоговое сообщение msgBox в VBA

Рассмотрим еще один вид диалоговых сообщений, которые содержат дополнительные кнопки «Да», «Нет», «Отмена»

Переходим к коду нашего макроса «Hello» и дописываем к команде **msgbox** следующее:

MsgBox "Hello, World!", vbYesNoCancel, "Мой макрос"

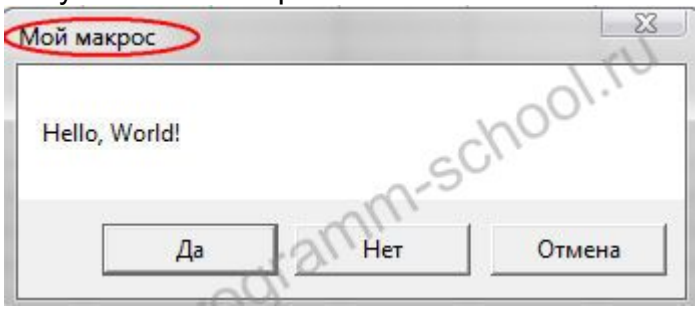
Обратите внимание, когда вы поставите запятую после «Hello, World!», у Вас должен появиться список всех доступных типов диалоговых окон



Поэкспериментируйте с каждым для понимания.

Как Вы заметили, я добавил еще один параметр к команде msgBox – “Мой макрос”. Это подпись нашего окна. Этот параметр не обязателен, но я рекомендую все диалоги подписывать наименованием своей разработки или иной информацией.

Запустим наш макрос



Теперь у нас открылась совсем иная форма сообщения.

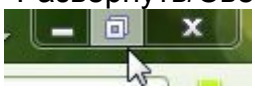
На этом пока все. Следите за выходом статей и уроков. Если возникли вопросы, с удовольствием отвечу, а если будет необходимо, напишу статью с пояснением

Создание формы в Excel на VBA

Опубликовал Deys в ср, 29/05/2013 - 22:11

Для чего нужны формы в VBA?

Форма это некий контейнер, в котором размещаются различные объекты и элементы управления, из которых создается интерфейс приложения, т.е. проще говоря, форма - это холст, на котором разработчик рисует лицо своей программы. С формами Вы сталкиваетесь постоянно. Например, окно браузера, в котором Вы читаете эту страницу, так же является формой с элементами управления и отображения информации. Характерный набор кнопок для формы это "Свернуть", "Развернуть/Свернуть окно", "Закрыть".

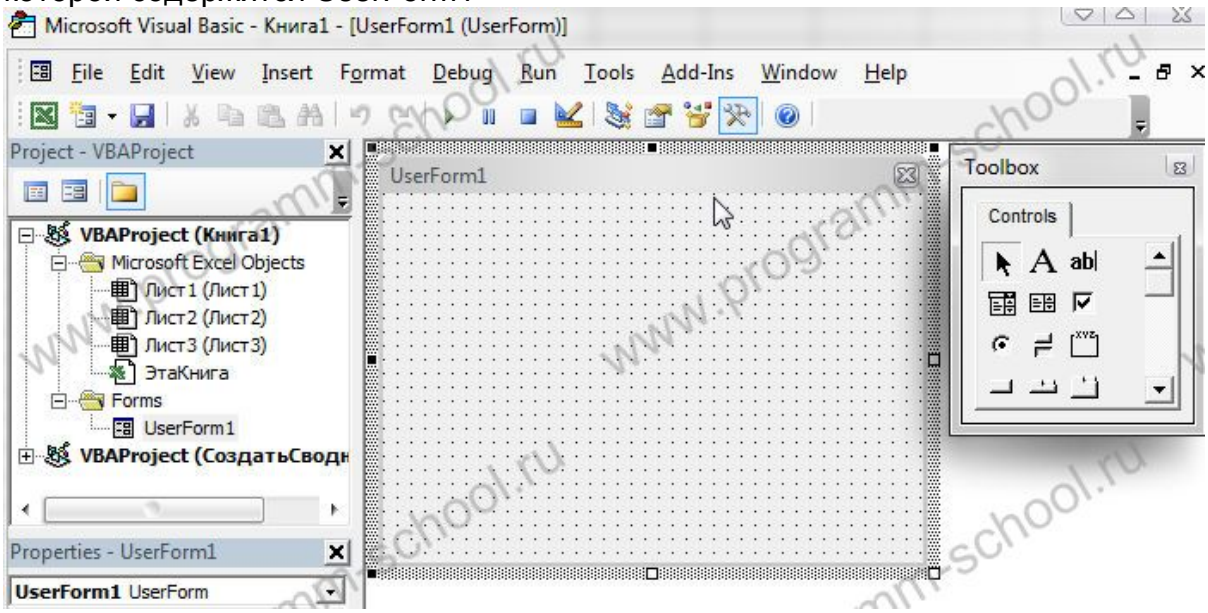


В операционных системах (например Windows, MacOS, Linux) содержатся библиотеки, в которых уже заложены функции построения стандартных форм и объектов управления, что значительно упрощает разработку интерфейса большинства прикладных приложении. Такой подход позволяет сделать приложения универсальными в плане переноса на другие рабочие станции (ПК) и легче т.к. нет необходимости "носить" все библиотеки с собой. Эти функции используются различными средами разработки.

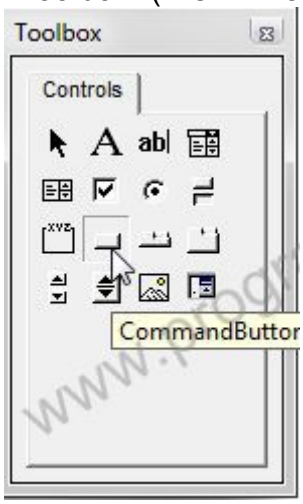
В VBA Excel формы позволяют организовать полноценный интерфейс для взаимодействия с пользователем, конечно, имеется много ограничений и неудобств.

Перейдем теперь к практической части **создания и конструирования форм**.

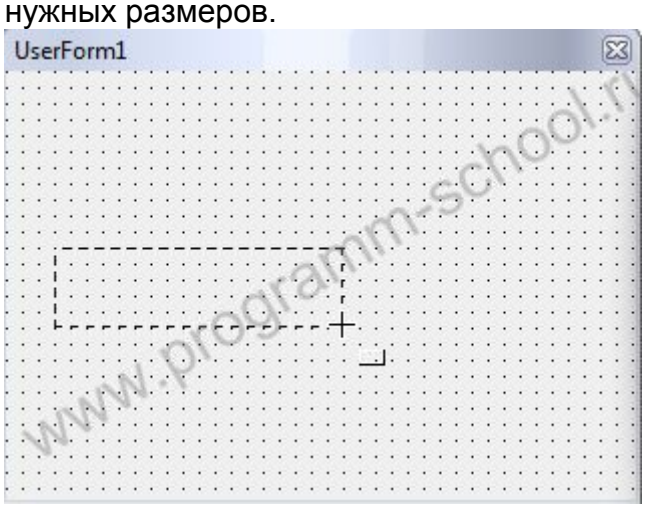
Добавляются формы в VBA просто, открываем редактор Visual Basic (**Alt+F11**), в главном меню редактора **Insert – UserForm.** После чего должна появиться форма, а в структуре книги (окно **View - Project Explorer**) появится папка "*Forms*" в которой содержится UserForm1



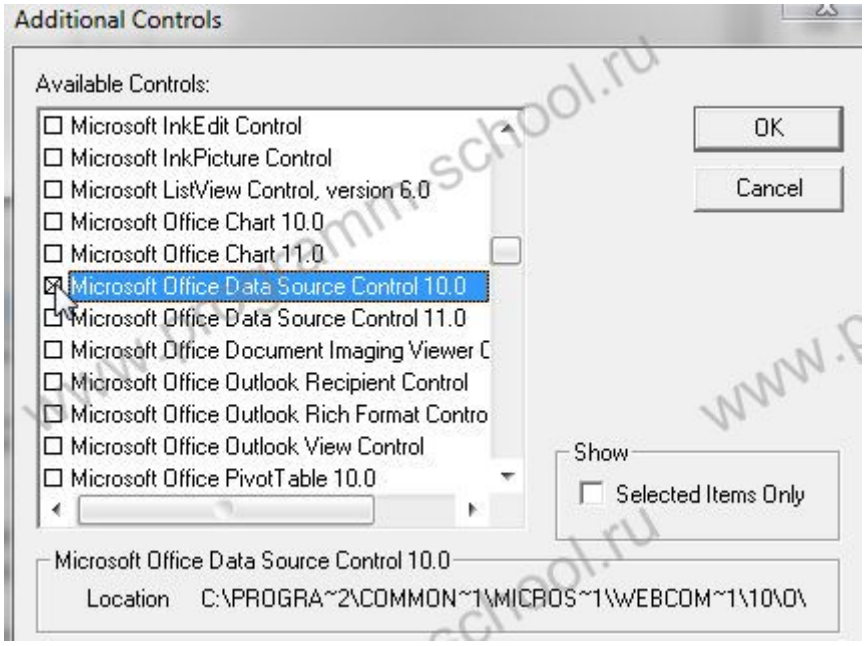
Конструируется форма в VBA очень просто, выбираем на панели объектов "Toolbox" (**View - Toolbox**) нужный объект, например "CommandButton"



переходим на форму, жмем ЛКМ (Левая кнопка мыши) и не отпуска тянем до нужных размеров.



Toolbox (Инструменты) - это панель содержащая необходимые для разработки интерфейса объекты (Кнопки, Метки, полосы прокрутки и т.д.). В панель Toolbox можно добавлять новые объекты, для этого щелкните ПКМ по свободной области панели и выберите из контекста "*Additional Controls...*". В открывшемся окне выбираете необходимые компоненты.



Учтите, что наборы компонентов могут отличаться на различных ПК и соответственно, приложения, написанные в Excel, не будут работать или будут работать, но с ошибками. Обратите внимание, если выбрать какой-либо объект, например "Кнопку", то в окне "Properties" (**F4**), появится список доступных свойств у данного объекта. Это очень удобно, когда необходимо ввести, например метку, поменять цвет или задать более точный размер и т.д. Поэкспериментируйте с созданием/изменением объектов и самой формой.

Как показать (запустить) форму на VBA?

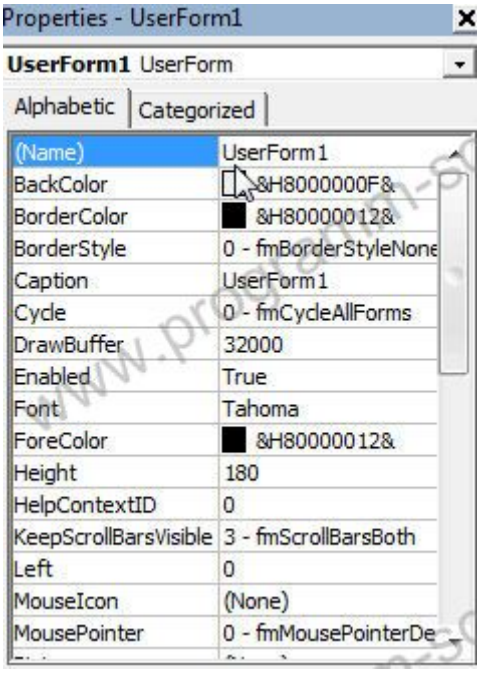
После того как форма была создана, ее необходимо при запуске макроса открыть. Как это сделать?

Все очень просто. Создайте в книге модуль (как это сделать, описано в предыдущей статье) следующего содержания:

```
Sub ЗапускФормы()
    UserForm1.show
End Sub
```

Запускаете макрос "ЗапускФормы".

Команда UserForm1.show - дословно UserForm1.Показать, где UserForm1 это имя нашей формы установленное в поле Name. Show(Показать) - метод формы. Для скрытия форм используется метод Hide (Скрыть).



Типы данных в Visual Basic (VBA)

Опубликовал Deys в чт, 01/08/2013 - 00:14

Все типы в VB можно разделить на несколько групп:

- Целочисленные
- Вещественные
- Строковый (текстовый)
- Логический
- Дата
- Неопределенный
- Объект

Целочисленные типы данных

К этой группе относятся все данные только целого типа. Диапазон зависит от выбранного типа (см. таблицу).

Тип	Диапазон	Описание
Byte	от 0 до 255	Этот тип данных хранит положительные целые числа до 255. Занимает памяти 1 байт (8 бит).
Integer	от -32768 до 32767	Этот тип данных позволяет хранить как отрицательные, так и положительные целые числа. Требует памяти 2 байта (16 бит)
Long	от -2147483648 до 2147483647	Самый длинный целочисленный тип. Хранит как отрицательные, так и положительные целые числа. Памяти требует 4 байта (32 бита)

Вещественные типы данных

К этой группе относятся данные содержащие дробную часть. Так же возможно использование этих типов для хранения целых чисел. Но целыми они будут только внешне, при этом памяти «кушать» больше.

Тип	Диапазон	Описание
Single	от $-3{,}402823*10^{38}$ до $-1{,}401298*10^{-45}$ и $1{,}401298*10^{-45}$ до $3{,}402823*10^{38}$	Числа с одинарной точностью. Требуют памяти 4 байта (32 бита)
Double	От $-1{,}79769313486232*10^{308}$ до $-4{,}94065645841247*10^{-324}$ и от $4{,}94065645841247*10^{-324}$ до $1{,}79769313486232*10^{308}$	Числа с двойной точностью. Требуют памяти 8 байт (64 бита)
Currency	от -922337203685477,5808 до 922337203685477,5807	Числа с фиксированной точностью. После запятой всегда 4 знака. Числа этого типа не имеют ошибок при округлении. Подходит для денежных вычислений. Требует памяти 8 байт (64 бита)

Строковый (текстовый) тип данных

Для описания переменных содержащих символы алфавита, знаки пунктуации, цифры и др. символы, используется тип **String**. Тип**String** позволяет хранить строки как фиксированной длины, так и переменной.

String переменной длины позволяет хранить текстовые данные от 0 до 2147483648 символов и требует памяти один байт на один символ т.е. при максимальном размере 2048мб.

String фиксированной длины позволяет хранить до 65536 символов т.е. 64Кб

Логический тип данных

К логическому типу относится тип ***Boolean***. Хранит всего два значения 0 и 1 (TRUE и FALSE). Требует 2 байта памяти. Любые логические операции используют данные этого типа.

Тип данных дата

Тип ***Date*** позволяет хранить дату в диапазоне с 1 января 100 года по 31 декабря 9999 и время от 0:00:00 до 23:59:59. Требуется памяти 8 байт.

Неопределенный тип данных

К неопределенному типу относится тип ***Variant.*** **Variant** это тип данных, который используется во всех переменных с необъявленным явно типом. Этот тип данных может хранить любой из вышеперечисленных типов за исключение типа **Object.** На первый взгляд этот тип может показаться более удобным, однако его рекомендуется использовать только в исключительных ситуациях. Данные этого типа обрабатываются значительно медленнее и занимают в разы больше памяти. Этот типа требует памяти 16 байт, в случае если хранится текст, то к 16 + 1 байт на каждый символ.

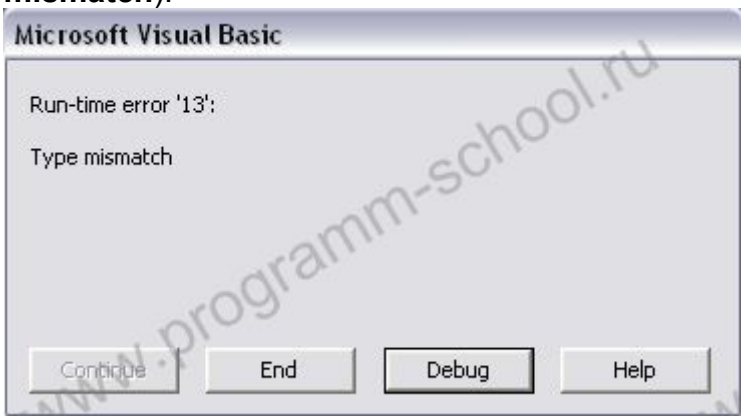
Тип данных объект

Тип ***Object*** используется для доступа к любому объекту известному в VBA. Переменная этого типа сохраняет адрес объекта. Использует память 4 байта.

Функции преобразования типов данных в VBA

Опубликовал Deys в чт, 01/08/2013 - 23:44

Часто в программировании возникает необходимость перевести один тип в другой (по доступным типам данных в VB читаем выше). Например, число в строку или строку в дату. Для перевода (преобразования) типов в VBA есть множество функций позволяющих это сделать. Но есть один момент, преобразовать возможно только тот тип или значение, которое подходит по формату нового типа. К примеру, если преобразовать строку "356" в целый тип, то на выходе мы получим число, но если в этой строке будет находиться символ, не относящийся к числу "356р", то преобразование завершится ошибкой несовпадения типов (**Type mismatch**).



В таблице ниже приведен полный список функций преобразования и тип, в который происходит преобразование:

Функция	Возвращает тип	Действие
CBool	Boolean	Преобразует значение в булевый тип
CByte	Byte	Преобразует значение в тип Byte
CCur	Currency	Преобразует значение в тип Currency
CDate	Date	Преобразует значение в Дату и Время
CDbl	Double	Преобразует значение в тип Double
CDec	Decimal	Преобразует значение в подтип Decimal типа Variant
CInt	Integer	Преобразует значение в целый тип
CLng	Long	Преобразует значение в длинное целое
CSng	Single	Преобразует значение в тип Single
CStr	String	Преобразует значение в строку
CVar	Variant	Преобразует значение в тип Variant

Использование функций преобразования типов

Работать с функциями преобразования легко. К примеру, необходимо преобразовать число 4568 в строку:

```
Dim OutStr as String
OutStr = CStr(4568)
```

Преобразование строки или числа в булевый тип

```
Dim OutBool as Boolean
'Функция вернет значение True
OutBool = CBool(1)
OutBool = CBool ("TRUE")
```

Преобразование строки или числа в дату и время

```
Dim OutDate as Date
OutDate = CDate("25/06/03 23:35")
OutDate = CDate("25.06.03")
OutDate = CDate("37797,9826388889")
```

Примечание: *Функции преобразования можно использовать непосредственно в выражении избегая создания лишних переменных.Например: x=y+CInt("456")+z*

Работа с условием If в VBA

Опубликовал Deys в чт, 13/06/2013 - 23:04

Условный оператор **IF** является основной частью любого языка программирования. Без него не обойтись при написании даже небольшой программы, в которой необходимо принять некоторое решение. Синтаксис конструкции If следующий:

If *условие* **Then** [*Команда* 1] [**Else** *Команда* 2]

Если перевести, то получается: **Если** условие **Тогда** *Команда* 1 **Иначе** *Команда* 2

Т.е. если условие истинно тогда выполняется некоторая *Команда* (*Команды*) иначе выполняются другие *Команды.*В этом варианте конструкции **IF** будет выполнено только одна *Команда*. **Else** можно пропустить.

Примечание*: При такой форме условия в Visual Basic после ключевого слова* ***Then*** *обязательно должна идти команда, а так же слова* ***Then*** *и* ***Else*** *должны находиться на той же строке что и* ***IF****, иначе интерпретатор выдаст ошибку. Если для удобства восприятия необходимо* **Команду 1** *перенести на новую строку, то необходимо воспользоваться символом "_" после* **Then**.

If *условие* **Then** _
[*Команда* 1] _
[**Else** *Команда* 2]

При таком варианте использования условия будет выполнено только одно действие. Если необходимо выполнить множество действий после **Then** или **Else,** то воспользуйтесь следующим вариантом написания условия:

If *условие* **Then**
[*Команда 1*]
[*Команда 2*]
...
[**Else**]
[*Команда 3*]
[*Команда 4*]
End If

Ключевое слово **Else** можно так же, как и в первом варианте не использовать, если нет необходимости.

И третий вариант конструкции, при котором происходит проверка условия, если первое условие не выполнено

If *условие 1* **Then**
[*Команда 1*]
[*Команда 2*]
...
[**ElseIf** *условие 2* **Then**
[*Команда 3*]
[*Команда 4*]
[**Else**
[*Команда 5*]
[*Команда 6*]
End If

В условиях также можно использовать логическое И (**And**), ИЛИ(**Or**) и отрицание НЕ (**Not**).

Рассмотрим несколько примеров использования выше перечисленных конструкций.

Пример 1

```
If a=b Then msgbox "a равняется b" Else msgbox "a не равно b"
```

Пример 2

В этом варианте **Else** не используем.

```
If a=b Then msgbox "a равняется b"
```

Пример 3

Используя «_» для интерпретатора Basic такая запись равносильна записи в Примере 1

```
If a=b Then _
   msgbox "a равняется b" _
Else msgbox "a не равно b"
```

Пример 4

```
If a=b Then
      msgbox "a равняется b"
      a = a+b
Else
      msgbox "a неравно b"
      c = b
End If
```

Пример 5

```
If a=b Then
      msgbox "a равняется b"
ElseIf a>b Then
      msgbox "a больше b"
```

```
Else
    msgbox "b больше a"
End If
```

Работа с циклом For в VBA

Опубликовал Deys в пн, 10/06/2013 - 21:47

В этом уроке будет рассмотрена работа с циклом **For** в VBA. Пример работы с циклом For, так же будет продемонстрирован пример создания формул в Excel с помощью макросов.

Цикл **For** работает по принципу счетчика. **For** применяется в тех случаях, когда необходимо повторить некоторые действия заранее известное кол-во раз. Например, цикл **For** часто используется при чтении массивов.

Цикл **For** имеет следующий синтаксис:

For *счетчик* **=** *начало цикла* **To** *конец цикла* [**Step** *шаг*]
группа операторов, команд и т.д.
    Exit For
Next *счетчик*

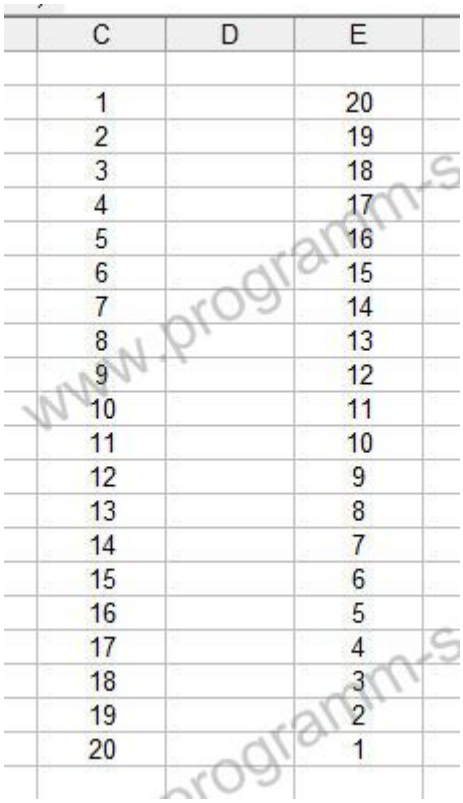
где,

- *"счетчик"* - переменная, которая изменяется на указанный *"шаг".* Если шаг не указан, то по умолчанию берется единица.
- *"начало цикла"*, "*конец цикла"* - числа или переменные указывающие нижний предел счетчика и верхний. Остановка цикла происходит тогда, когда *"счетчик"* > "*конец цикла" (*или, если цикл обратный, т.е. с шагом -1, то *"счетчик"* < "*конец цикла").*
- **Exit For –** команда принудительной остановки цикла. Применяется в тех случаях, когда произошло некоторое событие, после которого необходимо остановить выполнение команд в цикле, или для предотвращения возникновения ошибки.

Рассмотрим пару примеров использования цикла **For**. В дальнейшем, с этим циклом будем встречаться довольно часто.

Пример 1

Даны два столбца C и E заполненные числами:



C	D	E
1		20
2		19
3		18
4		17
5		16
6		15
7		14
8		13
9		12
10		11
11		10
12		9
13		8
14		7
15		6
16		5
17		4
18		3
19		2
20		1

Необходимо сложить числа в столбце C с числами столбца E следующим образом:
C2+E21, C3+E20, ..., C21+E2. Результат вывести в столбец D в виде формулы т.е. содержание ячейки результата должно быть "=C2+E21".
Код макроса выглядит следующим образом (куда прописывать код читаем здесь):

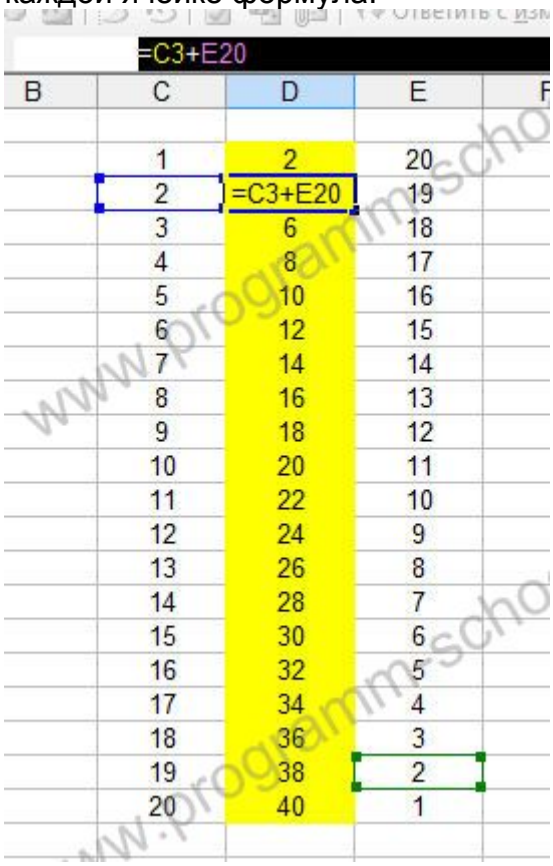
```
Sub Цикл_For()
'константа указывающая предел цикла т.е. до какого значения циклу бежать
Const n = 21
   For i = 2 To n
   ' создаем строку формулу и сохраняем ее в ячейку
   Cells(i, 4) = "=C" & CStr(i) & "+E" & CStr((n - i) + 2)
   ' продолжение когда выполняющегося в цикле
  Next i
   ' остальной код программы
End Sub
```

Разбираем написанный код:

- **Const** n = 21 - описание константы n со значением 21, т.е. число строк по которому необходимо пробежаться циклу **For**;
- **For** i = 2 **To** n - i счетчик который будет изменяться на 1 с каждым проходом цикла. Счетчик начинается с 2 и заканчивается когда i>n;
- Cells(i, 4) - ячейка выделенного листа, i номер строки, 4 -номер столбца в который выводится результат. Обратите внимание, наш счетчик i указывает номер строки листа Excel;

- **Next i -** оператор закрытия цикла и перевода указателя к **For.** Все что находится между **For** и **Next** выполняется в цикле;
- **CStr -** функция преобразующая число в текст.

Ячейке мы присваиваем формулу созданную следующим образом "=C" & **CStr**(i) & "+E" & **CStr**((n - i) + 2). Знак & - "склеивание" символов, строк. В результате у нас получится формула "=Cn+E((n - i) + 2)" где n = 21, i - счетчик.
Страшно? Это только кажется :)
Все. После выполнения макроса мы получим следующий столбец (выделен), а в каждой ячейке формула:

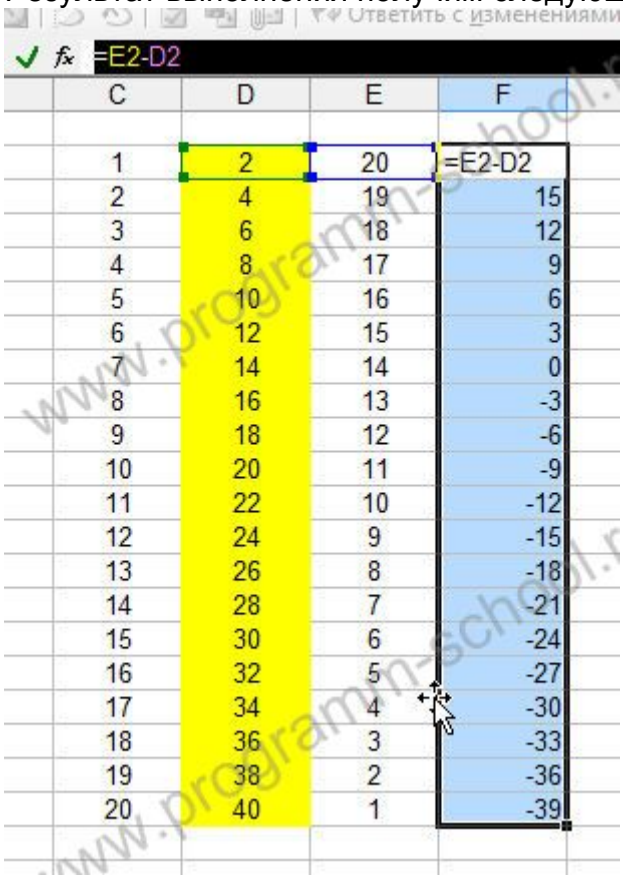


Пример 2

Теперь рассмотрим цикл с указанным шагом. После расчета прошлого макроса мы получили три столбца, теперь нам необходимо из столбца E вычесть D, в столбец F вывести формулы вычитания. Код макроса следующий:

```
Sub Цикл_For_с_шагом()
 Const n = 21
   For i = n To 2 Step -1
     Cells(i, 6) = "=E" & CStr(i) & "-D" & CStr(i)
   Next i
End Sub
```

В данном случае все тоже самое, только цикл теперь "бежит" не от 2, а от 21 до 2 с шагом (Step) -1.
Результат выполнения получим следующий:



Цикл **For**, в VBA, является не единственным циклом. В дальнейшем будут рассмотрены еще пара вариантов циклов, без которых не обойтись при написании макрокоманд в Excel.

Работа с циклами While и Until в VBA

Опубликовал Deys в сб, 29/06/2013 - 00:47

Выше мы рассмотрели цикл, который работает по принципу счетчика т.е. выполняется от a до n, с определенным шагом. Такой цикл подходит в тех ситуациях, когда известен интервал(кол-во проходов). Но что делать если цикл должен выполнятся столько раз, пока не наступит определенная ситуация, или наоборот, пока не наступает определенная ситуация? Для этого в Visual Basic (и не только в БЕЙСИКе) существуют циклы с условием - условные циклы.

Что такое условные циклы?

Это циклы - работа которых продолжается или завершается по указанному условию. Запись условия аналогична записи условного оператора IF. Условные циклы могут содержать одно или более условие, используя логические операторы: И(AND), ИЛИ(OR) и НЕ(NOT).

Условный цикл Do While.

Синтаксис записи условного цикла While выглядит следующим образом:

с предусловием

```
Do [While условие]
   [действия]
Loop
```

с постусловием

```
Do
   [действия]
Loop [While условие]
```

Разница в этих двух записях в том, что во втором случае, цикл будет выполнен хотя бы раз. Для принудительной остановки цикла можно воспользоваться командой **Exit Do.** Цикл **Do While** будет выполнятся до тех пор, пока заданное условие не выполняется.

Рассмотрим пример, в котором происходит подсчет ячеек столбца А до тех пор, пока не встретится пустая ячейка.

```
Public Sub Test1()
Dim i As Integer
 i = 1
   Do While Not IsEmpty(Cells(i, 1))
      i = i + 1
   Loop
 MsgBox i - 1
End Sub
```

Функция **IsEmpty** определяет, пуста ли ячейка.

Запись с постусловием и добавлением еще одного условия "*и пока ячейка не содержит 2*". Если в столбце встретится пустая ячейка или со значением 2, то произойдет остановка цикла, хотя последующие ячейки не пусты.

```
Public Sub Test2()
Dim i As Integer
 i = 1
   Do
      i = i + 1
   Loop While Not IsEmpty(Cells(i, 1)) And Cells(i, 1) <> 2
 MsgBox i - 1
End Sub
```

Условный цикл Do Until.

В VBA имеется еще один условный цикл **Do Until...Loop**. Отличие от **Do While** в том, что выполнение цикла будет до тех пор, пока условие выполняется.
Синтаксис записи цикла **Do Until...Loop:**
С предусловием
Do [**Until** условие]
[действия]
Loop

С постусловием
Do
[действия]
Loop [**Until** условие]
Для принудительной остановки цикла так же используется команда **Exit Do**.

Рассмотрим реализацию примера выше, с помощью **Do Until**.

```
Public Sub Test3()
Dim i As Integer
 i = 1
   Do Until IsEmpty(Cells(i, 1))
      i = i + 1
   Loop
 MsgBox i - 1
End Sub
```

Как видите, в условии отсутствует **Not**.

Вот и все.

П.С.: *При работе с условными циклами, будьте внимательны! Зацикливание (бесконечное выполнение) при неправильно заданном условии для этих циклов частое явление. В случае, если произошло зацикливание*, *воспользуйтесь сочетанием клавиш прерывания CTRL+Pause(Break). Рекомендую так же, перед запуском цикла сохранить проект.*

Пример работы с оператором Select..Case в VBA

Опубликовал Deys в сб, 15/06/2013 - 01:55

В прошлой статье я рассказал о работе с условным оператором IF в VBA. Но что делать, если условий в задаче возникает очень много? Использовать конструкцию If многократно неудобно, и это усложняет читабельность кода. Для таких целей в VBA предусмотрена еще одна очень полезная конструкция - управляющий оператор **Select...Case.**

Select...Case позволяет по определенному значению, диапазону или условию выполнить разные действия. Например:

если a = 5 то выполнить *действие 1*

если a в диапазоне между 7 и 9 то *действие 2*

если a не подходит ни одно условие то *действие 3*

Это можно записать условным оператором **If,** получается следующее:

```
If a = 5 Then
   действие 1
 elseif (a > 7 And a < 9) Or (a = 7 Or a = 9) then
   действие 2
Else действие 3
End If
```

Сложновато, не правда ли!? А теперь представьте, если условий будет гораздо больше.

Теперь запишем это же, но с помощью конструкции **Select**:

```
Select Case a
    Case 5
        действие 1
    Case 7 To 9
        действие 2
    Case Else
        действие 3
End Select
```

Код получился немного длиннее, но намного понятнее и читаемый. Теперь рассмотрим конструкцию:

- **Select Case** a - ключевые слова конструкции, после которых идет переменная "a" которую необходимо проверить. Тип переменной может быть, целым, вещественным, строковым, символьным, логическим;
- **Case -** ключевое слово, после которого указываются варианты условия;
- **Case** 7 **To** 9 - проверяет, входит ли "a" в диапазон от 7 до 9;
- **Case Else** - если "a" не подходит ни под одно условие, то выполняется ИНАЧЕ. Эту строку можно и не использовать;
- **End Select -** ключевые слова означающие завершение конструкции **Select.**

***Пример работы* Select Case**:

```
Sub Test()
  a = 5
    Select Case a
        Case 5: MsgBox "a=5"
        Case 7 To 9: MsgBox "a между 7 и 9"
           Case Else: MsgBox "a неподходит"
    End Select
End Sub
```

И еще один пример демонстрирующий задание условия в **Case**:

```
Sub Test()
  a = 3
   Select Case a
     Case Is > 5
          MsgBox "a больше 5"
     Case Is > 7
          MsgBox "a больше 7"
     Case Else
          MsgBox "Не подходит ни одно условие"
   End Select
End Sub
```

Обратите внимание, при указании условия ">", после **Case** добавляется ключевое слово **Is**.

Переменные и константы в VBA

Опубликовал Deys в сб, 31/08/2013 - 23:59

Основное назначение VBA это обработка данных. Некоторые данные могут храниться в объектах, например, диапазонах рабочих листов, а некоторые сохраняются в переменных.

Что такое переменные?

Переменная это некоторое выделенное пространство в памяти компьютера, которое может содержать данные разных типов – числовые, текстовые, логические и т.д. а так же результаты вычислений. Значение переменной присваивается с помощью знака равенства. Для удобства работы с переменными в языках программирования предусмотрена возможность задания человеко-понятных имен, к примеру, имена переменных rFIO, rBirthday. К именам переменных в VBA имеется ряд требований:

- Переменная может содержать латинские символы, числа. Использование пробела и точки в имени переменной недопустимо (вместо пробела программисты используют знак "_");
- Имя переменной должно начинаться с текстового символа (числа в начале не допустимы);
- Не допускается использование следующих символов: #, $, %, &, !. Эти символы в VBA зарезервированы за кратким указанием типа переменной без ее описания. К примеру: содержание символа $ (MyTxt$) в конце имени переменной указывает, что значение хранимое в этой переменной текстового типа;
- Недопустимо использование зарезервированных слов. Если все ж для понимания назначения переменной необходимо использование в имени зарезервированного слова, то можно добавить некоторый символ или символы, например: Date зарезервировано (тип дата), для использования можно добавить "My", после чего получим допустимое - MyDate;
- Имя переменной не должно быть длиннее 254 символов. Но я не думаю, что кому-то придет в голову для переменной задавать имя длиннее 15-20 символов, не говоря уже о 254;
- Язык VB не чувствителен к регистру, поэтому переменные MyVar и myvar для VBA одинаковы.

Пара советов по именованию переменных. Давайте имена переменным понятные, но в тоже время короткие т.к. с длинными именами сложнее работать. VBA позволяет описывать переменные в любой области кода, что в дальнейшем, при большом кол-ве строк программного кода может вызвать затруднение с поиском и определением типа. Из своего опыта советую не лениться описывать все переменные в начале процедуры или функции, определив, таким образом, некоторый блок переменных. Можно также оставить комментарии с подробным описанием назначения переменной.

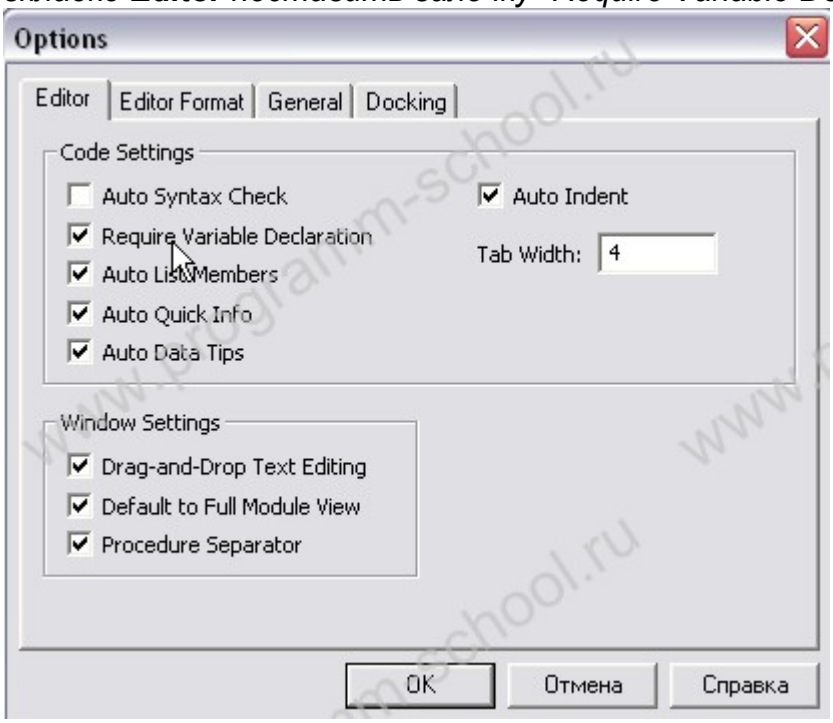
Описание переменных в VBA

Под описанием переменной подразумевается указание типа данных. В VBA переменные можно указывать явным и неявным образом. Не явным образом означает, что Вы можете в любом месте кода указать имя переменной и начать с ней работать, в таком случае тип этой переменной принимается как Variant. Такой способ удобен, но не рекомендуется т.к. может возникнуть путаница и как следствие, ошибки при вычислениях. Явное описание осуществляется после ключевого слова **Dim** [*имя переменной*] **As** [*Тип*], например: **Dim**MyInt **As** Integer. В случае такого описания переменная MyInt будет хранить в памяти только значения целого типа.

Обязательное объявление всех переменных

Как уже говорилось выше переменные в VBA можно объявить, а можно и не объявлять. Со своей стороны я рекомендую объявлять каждую переменную, используемую в программе. Это позволить сделать код более понятным, дисциплинирует, и в крупных разработках позволит сэкономить память и увеличить быстродействие при обработке данных. Для того чтоб в VBA включить обязательное объявление всех переменных необходимо в начале модуля добавить строку: **Option Explicit**. После, на каждой не объявленной переменной будет происходить остановка программы, и отображаться ошибка до тех пор, пока всем переменным не будет присвоен тип в разделе **Dim**.

*Примечание: Для того чтоб в VBE (Visual Basic Editor) оператор **Option Explicit** вставлялся автоматически в каждый новый модуль, необходимо в настройках редактора VBE активировать данную опцию: **Tools-Options...**, на вкладке **Editor** поставить галочку "Require Variable Declaration"*



Константы в VBA

Константой называется значение, хранимое в памяти, которое в процессе работы программы не изменяется. Константы используются в тех случаях, когда в коде программы используется один и тот же параметр, число, строка и т.д. Для того чтоб обеспечить удобство изменения этих параметров при необходимости, в начале кода программы описываются константы и присваивается значение. Константы объявляются с помощью оператора **Const**. Например:

Const MyIntConst **as** Integer = 8

Const MyTxtConst **as** String = "Константа"

или без указания типа:

Const MyConst = 4,55

в этом случае константа типа **Variant**.

Внимание!!! *В случае если при работе программы Вы захотите изменить значение константы, программа завершится ошибкой.Поэтому будьте внимательны. Так же, не допускается совпадение имен констант и переменных. Требования к именам констант такие же, как и к переменным.*

Закрепим все вышесказанное примером небольшого макроса, который рассчитает функцию y = x+i*2, где x=i/7, 0<i<100 шаг 1.

Как видно из функции **y, x, i** это переменные. Так как при расчетах используется деление, то **y, x** будут содержать значения вещественного типа. Переменная **i** это счетчик с шагом 1. 2 постоянное значение, т.е. константа. Результат **y** будет выведен в диалоговом сообщении **msgBox**.

Создадим в редакторе VBE новый модуль и добавим туда следующий код:

```
Option Explicit

Sub VarExample()
 Const MyConst As Byte = 2
 Dim Y As Single
 Dim X As Single
 Dim i As Integer

   For i = 1 To 100
     X = i / 7
     Y = X + i * MyConst
   Next i
 MsgBox "Результат равен " & Y
End Sub
```

Запускаем наш макрос. В результате выполнения данной процедуры мы получим Y равное 214,2857

Автор: Deys